(72) Inventors: BRANDT, Toni; Kvarnhagsvägen 135, S–145 60 Norsborg (SE). HÄGGBLAD, Pär; Djäkneg. 9:650, S–754 23 Uppsala (SE). ANDERSON, Jonas; Hjorvägen 12, S–178 32 Ekerö (SE). JÄNDEL, Magnus; Vårvägen 10, S–194 60 Upplands Väsby (SE). KARLSSON, Krister; Lindhagensgatan 69, S–112 43 Stockholm (SE). KARLSSON, Roland; Drakenbergsg. 3 IV, S–117 21 Stockholm (SE). LÖNNBERG, Emanuel; Djäkneg. 19:510, S–754 23 Uppsala (SE). OSBORNE, Stuart; Cedergrensv. 27, S–122 63 Hägersten (SE). STENHOFF, Martin; Solhems Hagväg 90, S–163 56 Spånga (SE).

(74) Agents: BERG, S., A. et al.; Albihns Patentbyrå Stockholm AB, P.O. Box 5581, S–114 85 Stockholm (SE).

(54) Title: METHOD AND APPARATUS IN A COMMUNICATION NETWORK
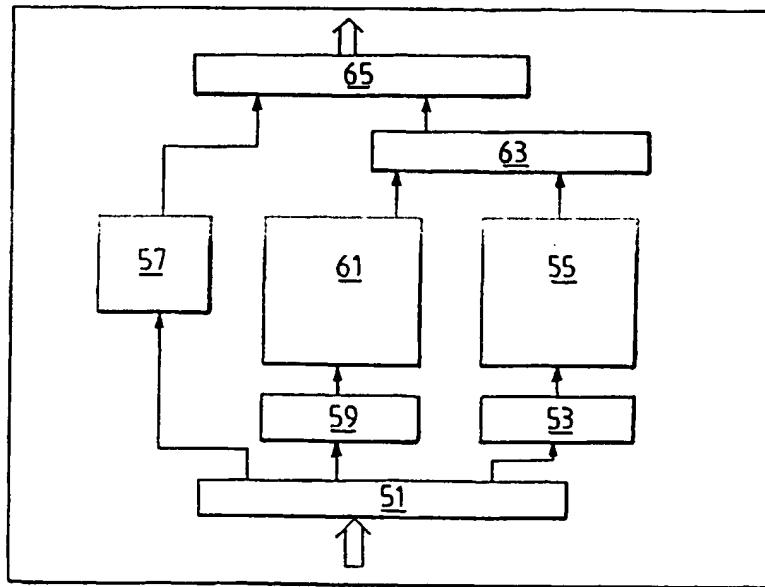
(57) Abstract

The real–time performance in communications networks is improved, especially between a large number of participants, by means of a server unit comprising receiving means for receiving information from at least a client unit, said information comprising at least part of the state information about a distributed interactive application, said server unit comprising: state information storing means for storing application state information received from at least one of said client unit, transmission means for forwarding the state information received from said client to at least one other node in the network, and for transmitting at least part of the information stored in said state information storing means to said at least one client. In this way the whole state of the application can be kept in one or more units in the network, which removes the need for each client to store the entire state, thereby reducing memory and banwidth requirements for each client.

Method and Apparatus in a Communication Network

Technical Field

The present invention relates to communications networks and in particular to the
real-time communication between several users of an application in such a network.

Description of Related Art

Multi-user communications applications are currently being developed, for example
for multi-user games. Such games may be played by a number of users connected to
the same LAN or even to different parts of larger networks, such as the Internet. For
such applications, three main synchronization models are used: client-server syn-
chronization, peer-to-peer synchronization or broadcast synchronization. Some
multi-user games on the Internet today involve up to 100 000 players, of which sev-
eral thousand may be playing at the same time. All synchronization methods used
today cause rather long delays. Therefore, the multi-user games played over the
Internet today are games in which the speed is not critical. For example, fast action
games, such as car racing games or battle games, in which the user must react to
what is happening within fractions of a second cannot be played with acceptable
quality using any of the standard synchronization methods.

Dial-in networks typically use client-server synchronization. In this case, a server
receives data from all players. It has to find out what information each player needs
and send it. The central server is an obvious bottleneck in the system. Central client-
server games can support 2 – 250 players. The higher number refers to servers
where the update rate is as low as 2 Hz. The central server adds latency both be-
cause of the increased transport distance and the processing and scheduling delay in
the server. Consider e.g. a situation where the clients are on the U.S. West coast
while the server is on the U.S. East coast. About 80 ms of transport latency and at
least several 10 ms of processing latency are caused by the client-server mode of
operation.

Therefore, client-server synchronization is not feasible for real-time applications with large numbers of players in which the delay is critical.

Peer-to-peer synchronization means that all clients send application data directly to all other clients. This model is frequently used for games that are played over the public Internet. Game developers provide a free application lobby server where players meet for setting up games and for joining ongoing games. Once a game has started it is played in the peer-to-peer mode without drawing any resources from the game developer's site.

Peer-to-peer synchronization has obvious scaleability problems since the network load is proportional to the square of the number of players; the client access band-width and CPU power requirement are proportional to the number of players. The payload in each packet is 10 – 40 bytes, which means that the protocol overhead usually is more than 50%. Small game packets give a large protocol overhead. With header compression methods, however, this overhead can be reduced significantly. Peer-to-peer games over the Internet suffer from unpredictable delays and frequent collapses because of loss of synchronization.

It follows that peer-to-peer synchronization, too, is best adapted to small networks, with communication over relatively short distances and between a limited number of users.

Broadcast synchronization can be used by a limited number of players on a Local Area Network (LAN) but is not suitable for larger networks.

Because of the restrictions posed i.a. by the synchronization methods, multi-participant communications with good real-time characteristics is not possible with current technology.

With multicast the information that has to be transmitted in the network can be sig-nificantly reduced, in that information only has to be transmitted once to each mul-

ticast server, instead of once to each user. Each multicast server then copies the in-
formation and transmits it to the users connected to it. A selection of the informa-
tion each user needs to receive may be made. This may reduce traffic both between
servers and between each server and its users. Such a solution for use in gaming is

5   disclosed in US 5 841 980.

In the system disclosed in US 5 841 980 each user only has the information related
to the part of the game which is relevant to the user in the current situation, e.g. in-
formation about the geographical area he is in. If the circumstances change, for ex-

10   ample, if the user moves to another room, information that is important in the new
circumstances will not be available to the user.

Object of the Invention
It is an object of the present invention to improve the real-time performance in

15   communications networks, especially for collaborative communication between a
large number of participants.

Summary of the Invention
This object is achieved according to the invention by a server unit for use in a

20   communications network, said server unit comprising receiving means for receiving
information from at least a first client unit, said information comprising at least part
of the state information about a distributed interactive application, said server unit
being characterized in that it comprises

-    state information storing means for storing application state information com-

25   prised in the information received through the receiving means from at least one
of said first and second client units,

-    first transmission means for forwarding the state information received from said
at least first client unit to at least one other node in the network,

-    second transmission means for transmitting at least part of the information stored

30   in said state information storing means to said at least one client unit.

This enables keeping the whole state of the application in one or more units in the network, which removes the need for each client to store the entire state, thereby reducing the need for memory capacity in each client and also the bandwidth needed for communication with each client. If each part of the state of the applica-

5   tion is stored in more than one network node or client, a back-up facility is achieved.

The object is also achieved according to the invention by a client unit for use in a terminal in a communications network, comprising an application software for a

10   distributed interactive application client unit being characterized in that it comprises

- at least one input means for reading an input from said terminal, said input con-
  stituting at least application state information for the distributed interactive appli-
  cation

- transmission means for transmitting application state information to an applica-

15   tion access server

- receiving means for receiving at least application state information for the dis-
  tributed interactive application from said application access server

- means for displaying said state information.

20   The method and apparatus according to the invention are particularly useful for distributed interactive applications, especially involving real-time communication.

Preferably, the receiving means of said server means is adapted to

- receive subscription information from at least said first client, said subscription

25   information identifying objects of the distributed interactive application about which said first client wishes receive information,

- transmit information to said at least one client unit in dependence of the sub-
  scription information received from that client unit,

- said server unit further comprising at least one client subscription list for storing

30   said subscription information.

Accordingly, said client unit further comprises means for setting subscription information for specifying at least one object of the distributed interactive application from which available information should be received and means for transmitting said subscription information to said application access server.

5

This reduces the amount of information to be transmitted to each user, thereby reducing the transmission delay and also helping each user analyze the information, since only the most important information for a particular user is shown to that user. Each client is able to decide for itself what is important.

10

In a preferred embodiment, the receiving means is adapted to receive urgency information relating to said first client from at least a second client. The urgency information may be transmitted to the first client, or application state information regarding said second client may be forwarded to at least said first client in dependence of said urgency information. The urgency information may also be used to indicate that certain application state information should not be forwarded to said first client. The urgency information may be used to change a client subscription.

15

In said preferred embodiment the client unit further comprises means for setting urgency data for specifying at least one other client that should receive state information from the client as soon as possible.

20

Advantageously, the transmission means of the client unit for transmitting application state information is adapted to arrange information in object information packets, each packet relating to one object constituting part of the application, before transmitting the packets to the server unit, and said receiving means is adapted to extract information from packets received from the server unit.

25

The application access server system is independent of the application and can therefore support a wide range of different applications.

30

Application access server units working together could communicate over a network, which preferably offers reserved or managed transmission capacity. The aggregated bandwidth requirement of an application can be estimated by the network management system and sufficient network resources can be allocated to the re-

5   served network that is connecting the application access server units. New applications are allowed only if resources are available. The network management also controls the resending and duplication policy of the application access server units. Multicasting and resource reservation protocols can be used on the aggregated streams between the application access server units. The advantage of this system is

10   that resource reservation on the player level is not necessary. Application clients can usually handle occasional lost packets if the overall statistical performance is good. Using the application access server according to the invention means that a game client never will lose synchronization permanently since the application access server units always maintain the game state.

15

Brief Description of the Drawings

The apparatus and method according to the invention will be described in more detail in the following, with reference to the appended drawings, in which:

Figure 1 illustrates an embodiment of a network according to the invention;

20   Figure 2 illustrates an application access server according to a first embodiment of the invention;

Figure 3 shows an embodiment of an application state record according to the invention;

Figure 4 shows an application client according to an embodiment of the invention;

25   Figure 5 illustrates the communication stacks used according to an embodiment the invention;

Figure 6 shows a second embodiment of the application access server according to the invention;

Figure 7 shows an application router used in the second embodiment of the applica-

30   tion access server;

Figure 8 is a schematic representation of a hierarchical structure of application access servers used in an embodiment of the invention.

Detailed Description of Embodiments

5   Figure 1 shows an embodiment of a communications network according to the invention. According to the invention, the network comprises at least one application access server. In Figure 1, a first 1, a second 3 and a third 5 application access server are shown, connected to each other through a network 7, for example, a reserved telecommunications network. It may also be any other type of network,

10   however, networks in which it is not possible to reserve network resources, such as the Internet, may provide a lower quality if heavily loaded. A number of clients 11, 12, 13, 14, 15 are also connected to the network 7. A client can connect to the appropriate application access server 1, 3 or 5 in any way known in the art, as will be discussed in more detail below.

15

Clients can communicate with each other through the application access servers for the purpose of running a distributed interactive application, in which a large member of participants simultaneously affect and contribute to the state of the application, for example a real-time multi-user game. Between each client and the applica-

20   tion access server to which it is connected there will usually be a low capacity connection, such as a modem connection. Between two application access servers, on the other hand, high and variable network capacities will be available. The application access servers according to the invention are therefore adapted to handle the information about the state of the application in such a way as to reduce the amount

25   of information transmitted to each client, by transmitting to each client only the information that is most relevant to this client. Between the application access servers, the transmission capacity will not normally be a problem and therefore more information can be transmitted between application access servers.

One or more application lobby servers 21, of a kind known per se, may also be present in the network. These servers typically comprise functions that allow the clients to register for a particular type of service, handle billing functions, etc.

5   According to the invention, the software required for the application is found in the client 11, 12, 13, 14, 15. To use the communication functions according to the invention, therefore, the user must make sure he has the application software needed. The application software may be retrieved in any way known in the art, for example, downloaded from the Internet or installed from CD-ROM. If required for the
10   application he must also register with the application lobby server. Typically, the application lobby server 21 will provide an address, usually an IP address and port number of one or more application access servers 1, 3, 5, but this address may also be obtained in any other way.

15   The user connects to an application access server in the way common in the art, normally by entering the IP address of the application access server or a pool of application access servers. Several algorithms exist for selecting the appropriate one of a number of nodes in the network identified by the same address. The application access server receives information from the user and starts transmitting information
20   to the user, as will be described in more detail in the following. Instead of the client logging in to the application access server, the application lobby server can also transmit client network addresses to the application access servers.

The application is assumed to consist of a set of objects, which are controlled by the
25   participants. An object is an entity in the application that is controlled by a human participant or that is generated in a client and cannot be regenerated locally in each application client, e.g. controlled by a random generator or other unpredictable computing process. Application objects can be things that the participants will intuitively recognize as objects, or game figures, but they can also be background data
30   structures such as environment variables that are controlled by players. An application object is controlled by one or several players. Each object typically has a set of

properties and attributes that may be changed while the application is running. In a game these properties may be a character's strength and other capabilities, or the maximum speed of a car, and the attributes may be, for example, items collected by the character.

5

The information received from the user will normally comprise three different types of information, in addition to the signalling overhead: state information, subscription information, control data and urgency information. The state information is information about the object or objects that are controlled by the user, which should

10   be distributed to other users of the application. The subscription information lists the client's priorities regarding different application objects provided by the application access servers. For example, information regarding a certain group of objects should be received at once it becomes available, or as often as possible, whereas information from another group of objects is not that urgent. Several priority levels

15   may be set, for example, in the form of acceptable delays or precedence in case of lack of bandwidth. The urgency information generated by a client may be used to override or change the subscriptions of another client so that the other client will receive information about objects that are not included in its subscription, or to make sure that the client does not receive certain information. All three types of in-

20   formation will be discussed in more detail below.

As an example, the application of a real-time game being played by a large number of users, will be described. Such games are played today with a limited number of players, located fairly near each other, for example in a local area network (LAN).

25   Played over a wide area network, such a game may in the future involve several thousands of players, typically distributed over a large game field, or virtual geographical area. With prior art technology this is not possible with an acceptable quality, as discussed above.

30   Each player is immediately affected only by the things that happen near him in the virtual game area. The farther away another player is, the less important changes to

this player will be. For example, a player may be involved in a fight with a first hostile player, a friendly player may be coming to rescue and a second hostile player may be trying to stop the friendly player. At the same time, other players will be doing things that may become interesting at a later time, but for the time being,

5   the player's main concern is to survive the fight. Therefore, the movements of the first hostile player must be displayed immediately. The movements of the friendly player and the second hostile player should also be given a high priority, while the things happening farther away should be given a low priority.

10   In the example discussed above, obviously, the configuration of the players will change so that at another time information from other players will have the highest priority, usually the players who are located near the user at any given time.

The application access server handles communication with application clients. It re-
15   ceives information about the actions of the client that affect the application state and other information. It also transmits application state information from other participants to the participants according to their subscriptions. Each application client sets subscriptions regarding which application information to receive and transmits information about these subscriptions to the application access server. The application
20   access server stores and updates a record of the subscriptions of the clients that are served by the application access server. The subscriptions of each client are used to determine which data is to be sent to this client. A client might also set subscriptions for another client. In one mode of operation, a master client might for example set the subscriptions of all other clients.

25

The application access server to which a client is connected is that client's local application access server, and the client is referred to as a local client of that application access server.

30   The application access server also keeps a client authority list. Each client has the right to change the state of one or more game objects. The client authority list has

one entry for each client where the identifiers of the objects that are controlled by the client are stored. The application access server checks the authority each time it receives object state information from a client. The game state will be updated only if an identifier found in the information packet matches one of the numbers in the
5   entry for the client in the authority list, otherwise the information is ignored. Object state information arriving from other application access server units is not verified since the sending application access server has already checked for authority.

Each application access server also communicates with other application access
10   server units, sending application state information and, optionally, aggregated sub-scriptions to them and receiving the same type of information from them. Usually, the information to be transmitted to the other application access servers is packaged according to an appropriate protocol and transmitted without being prioritized or sorted in any way. It would of course be possible to select the information to be
15   transmitted to each application access server in dependence of the users' subscrip-tions of its local clients. Aggregated subscriptions are the sum of all application object requests from clients belonging to a given application access server. For ex-ample, the application access server may use multicasting to other application ac-cess server units. This ensures that each application access server unit receives all
20   needed updates and that the game state is distributed efficiently over the application access server units.

Urgency data is set by a client if the data is important to another client but this other client is not aware of this and therefore cannot set his own subscriptions. The ur-
25   gency is therefore defined by the sending client and may be different for different receiving clients. The application access server alerts the receiving client if urgent data is waiting. The urgency information may also be used to prohibit the forward-ing of certain information to one or more clients.

30   Each application access server also stores and updates a complete or partial copy of the application state. Between them, the application access server units that are allo-

cated to a communication session must store the complete application state. In the simplest case, every application access server stores the complete application state, but it is also possible to split the information between the application access servers, with or without overlapping information. Such solutions require special software to

5   handle the distribution of the appropriate data to each application access server.

The application access server communicates with the application lobby server for setting up the application, adding and removing participants while the application is running and handling errors and network failures.

10

Figure 2 shows an application access server 50 according to an embodiment of the invention, with its functional entities. This application access server could be implemented in hardware or in software. In connection with Figure 2, only a general description is given. Appropriate formats and protocols will be discussed later.

15

The application access server receives data from each client in the network, and from other units, such as other application access servers, through input buffers 51. There is one input buffer for each client, remote application access server and application lobby server, although in Figure 2, for clarity, only one input buffer is

20   shown.

The actions of each local participant are sent as payload in application object packets that arrive in the input buffer of the application access server associated with this client. These application object packets update application objects that are con-

25   trolled by the participant. The application object packets are written into the application state record using a set of insertion rules.

The data received from the clients may comprise three types of data: data about the state of the application (state data), data about the subscriptions of the client (prior-

30   ity data) and other control information, such as requests for time reference, status of other clients and other information from the system. The state data is passed

through an application object packet pipeline 53, to an application state record 55. The application state record 55 keeps all the state data for all the relevant objects, as will be described in more detail in the following. The application state record also keeps the urgency data as described above.

5   Groups of clients may be defined, said group for example comprising the participants in a working group or, in the case of a game, in the same team. In this case, subscriptions and/or urgency data may also be specified for groups of users, not only for individual users.

10   Control data, especially data that does not have to be handled in real time, such as information on added or removed application objects, is passed from the input buffer 51 to the application access server control unit 57. In a hardware implementation the control unit 57 may be a microprocessor. Control data handled by the control unit could be related, for example, to the creation and destruction of players

15   and objects, and to groups of players and objects.

Subscriptions are passed through an application priority control protocol message pipeline 59 to a Client Priority List (CPL) 61 containing the subscriptions of each client that is supported by the application access server. The subscriptions can be

20   updated by the clients and by the application access server control unit. The control unit could decide, for example, to update the subscriptions of a client in response to an urgency message received from another client.

Several different prioritization strategies can be employed by the application clients

25   using the application access server system.
A simple method is that application clients send an enumerated list of objects to the CPL 61. In this case objects not on the list could be updated, for example, in a round-robin mode when all the objects on the list have been updated. If all objects in the list have been updated the remaining objects could be updated, for example in

30   a round robin mode. The subscriptions of a particular client comprises a list of all objects for which a subscription has been set by this client. In addition the list stored

in the application access server preferably comprises a flag for each object indicating if new information for the object has been received. The flag is used to determine whether or not information about a certain object needs to be sent to the client. When the information has been sent to the client, the flag is reset. When a new up-

5   date for an object is received by the application access server, the flag is set again.

The client subscriptions could also be defined with a time interval associated to each object. The client would then send a series of requests, for example of the format:

10   <object number>, <object priority>, <object update time interval>, <flags>. A simple list could simply have the format <object number>, <object priority>.

The application access server will try to send updates to the client so that each object is updated at least once during each time interval with the priority given according to the <object priority> field in the request. An infinity symbol could be

15   used so that some objects are not updated at all. A special "send only new complete object state" flag can be set. This option is used when the client wants to receive· updates for the object with long intervals (several seconds). It would be wasteful to send incremental updates if the wanted time resolution is longer than the state refresh interval.

20

An output pipeline 63 receives subscription information from the client priority list 61 and uses this information to search the application state record 55 for information to be transmitted to the client in question, using an appropriate algorithm as discussed above. The selected state information passes from the application state

25   record 55 through the output pipeline 63 to an output buffer 65 and from the output buffer to the client. The control unit 57 sends control messages to clients and remote servers, for example, for the purpose of synchronizing clocks or providing latency estimates to clients.

30   Although the application access server shown in Figure 2 is shown to have only one of each type of unit, preferably the application access server comprises one input

buffer 51, one application object packet input pipeline 53, one priority message
pipeline 55, and one output processing pipeline 63 and one output buffer 65 for
each client, and for each other server, etc. in the network with which the application
access server exchanges information. Alternatively, a group of clients or servers,

5   could share one input buffer 51, one application object packet input pipeline 53, one
priority message pipeline 55, and one output processing pipeline 63 and one output
buffer 65. The application access server also comprises one client priority list 61 for
each client, server, etc.

10   The data received from other application access servers comprises object state data
from other clients. It may also comprise priority information from these clients,
which is preferably aggregated in such a way that each object appears on the prior-
ity list only once. In this case, the information will be packaged so that there is one
priority list for each application access server, that is, each application access server

15   is treated like a client. The information received from another application access
server is treated in the same way as the information received from a client, except
that the input and output buffers 51, 65, also have to perform certain protocol han-
dling, as will be discussed below. Alternatively, no subscription information will be
received from other application access servers, in which case all state information

20   will be transmitted to these application access servers.

The application access server may also handle optional fair play modes where all
players are updated simultaneously (see below).

25   As mentioned above, one or more application lobby servers (21 in Figure 1), of a
kind known per se, may be present in the network. Application lobby servers are
responsible for updating the configuration data during the game providing the ap-
plication access server with data such as:

• IP addresses or other network addresses of new application access server units
30   that enter the game

- Updated lists of network addresses identifying the participants served by the application access server. This enables new participants to join an ongoing application.

- Updated lists of enumerated game objects. For each object it is specified which application access server is responsible for storing the state, if the state is distributed between the application access servers. It might also be specified who is authorized to update the state. This enables creation, destruction and change of control of game objects. Clients can also create and destroy game objects.

- A new complete or partial game state. This enables recovery after a pause or failure in the game when the application access server system has discarded the game state.

Figure 3 shows an example of an application state record used in the application access server according to the invention:

The state of each application object is stored in the application state record as a set of enumerated application object states AOS1 - AOS4. Each application object state consists of a series of application object packets AOP11- AOP13, AOP21-AOP24, AOP31 and AOP41-AOP42, respectively. An object packet is a container for object data. All data that is sent from an application to a remote application client or server is wrapped in an application object package so that it can be handled by the application access server system.

The information received by an application access server from all other application access servers to which it is connected is used to update the application object information in the application state record. Preferably, there are two types of application object packet: 1) a reference application object packet comprising all current data about an object, and 2) an incremental packet comprising only information about what has changed since a time that is given by the time stamp of the APO to which the incremental packet refers. The first AOP in an AOS must be a Reference

packet, which may be followed by a set of Incremental Packets, or by a reference packet. Some games will only generate reference packets.

The application state record also keeps a record of the application object packets

5   that need to be delivered to other application access server units. This may be done using a data structure new_client_data that can be implemented as a matrix new_client_data(i, k) where each element is a flag. The parameter i is a client number and the parameter k is an application access server number. The flag is set if the application object packet should be delivered to the external application access

10   server.

As an example of how application object states can be used to describe application objects, Figure 3 shows an application object state AOS1. This object might describe the position of a vehicle in a racing game. The position of vehicle (x1, y1) is

15   first sent as game payload in a reference application object packet AOP11 at game time t1. To save bandwidth, the relative change in the position (Δx2, Δy2) at game time t2 is transmitted as an incremental application object packet AOP12. The incremental application object packet AOP12 points to the reference application object packet AOP11 as reference. At game time t3 a new incremental position (Δx3,

20   Δy3) is sent in a third application object packet AOP13. The third application object packet AOP13 has the second application object packet AOP12 as reference. After having received all three application object packets AOP11, AOP12, AOP13, the client can calculate the position of the vehicle at time t3 according to (x1+Δx2+Δx3, y1+Δy2+Δy3).

25

When a new reference application object packet is received for a particular object, the application state record can delete all previous packets belonging to the object and just store the new reference packet. Note that the syntax and semantics of the game payload can only be understood by the game application running on the client

terminal. Applications should send reference packets frequently in order to avoid long interrupts caused by lost data.

An alternative way of coding the application would be to let the third application
5   object packet AOP13 at time t3 use the reference application object packet AOP11 as reference and state only the change of position, δx, δy relative to (x1, y1). This would save application access server memory since the second application object packet AOP12 at time t2 could be deleted as soon as the third application object packet AOP13 arrives. The client would now calculate the position of the vehicle at
10   time t3 according to (x1+δx3, y1+δy3). The application access server uses information in the application object packet header for determining if a previous application object packet is expired and can be deleted. In Figure 3 AOS 2 consists of four application object packets AOP21, AOP22, AOP23 and AOP24, where the first one AOP21 is a reference packet and the three following application object packets
15   AOP22, AOP23, AOP23 are incremental. The last increment has time stamp t8. AOS 2 thus describes the state of object 2 up to game time t8. The AOS 3 in Fig. 3 consists only of one reference application object packet AOP31, which is sufficient for describing the object.

20   Before allowing a new client to sign on to the application, the application access server may, according to a preferred embodiment, estimate the increased bandwidth needed and ensure that this bandwidth is available in the network. Methods for doing this are well known in the art. If no capacity problems are foreseen, this step will not be needed.

25

Figure 4 shows an embodiment of a computer on which a client according to the invention is running. The computer comprises a processing unit 101 in which programs are run, for example an application program 103 according to the invention. The processing unit also communicates with an application access server (not
30   shown) and possibly other units in the network, by means of communications soft-

ware 105. The application program 103 communicates with communication software 105 through a network application interface 107. The network application interface has functions for sending and receiving application data from the application program.

5

The computer also comprises a screen 109 for displaying data about the application, for example an overview of the part of the game that is of immediate interest to the participant. For inputting data to the application, for example, the computer may have a keyboard 111, a mouse 113 and/or a joystick 115 connected to it, by means

10   of which an object in the game may be moved, or other types of changes may be entered.

The client application 103 receives said input, processes it and displays the result of it on the screen 109, and/or by means of for example loudspeakers and/or haptic

15   display means. It also forwards application state data based on said input to the network application interface 107, from which it is forwarded to the application access server.

Through the communications software 105 and the network application interface

20   107 the application 103 also receives application state information concerning other objects from the game access server, processes it and displays the result on the screen 105.

In this embodiment the network application interface 107 comprises two parts: an

25   application programming interface 107A and an application access interface (AAI) 107B. This solution was chosen to enable the use of a standard program module, such as Microsoft DirectPlay, for implementing the network AAI 107B, as discussed below.

30   The application access interface (AAI) 107B is a software module in the client terminal. It is an intermediate module between the network interface and the network

API 107A. The application access interface 107B receives and terminates applica-
tion object packets and control messages, and removes application object packet
headers before the application object packet payload is passed to the network appli-
cation interface 107. It also translates control messages and passes them to the API

5   107A or handles them directly. The AAI 107B handles functions that are required
for the communication with the application access server to function but that are not
implemented in the client application program. Therefore, it may not be necessary
for clients that have been developed for use with an application access server ac-
cording to the invention. For example, the AAI can handle the clock that places

10   time stamps on the application object packages, if the client does not have functions
for this.

In the upstream direction, AAI 107B receives messages and objects from the API
107B. Data concerning application objects from the client is transformed into the

15   application object packet format. Application object packets are transmitted over
the communication link to the application access server.

The AAI 107B also generates upstream application control protocol messages, in
particular, application control protocol subscription messages. The information that

20   is needed for setting up subscriptions and other application control protocol mes-
sages must be extracted from the application via the network API and from urgency
list messages from the application access server.

A physical client e.g. a game console could be engaged in several games or host

25   several human players in the same game. Each physical application client can run
several logical application clients where a logical application client corresponds to
one instance of an application connected to one instance of the Application Access
Interface. Application clients in this document correspond to a logical application
client. A network address pointing to a logical application client could consist of the

30   IP address of the physical client combined with the port number of the application
access interface.

The best performance is achieved if the application access server system is considered when the application program is developed. Functions for determining subscriptions indicating the preferred order of receiving data can then be included in the application program. Messages could be directly directed to a co-player with high urgency if it is apparent from the situation in the game that the receiving player cannot predict the high priority of the message. Consider e.g. a situation in a game where player 1 is stalking player 2. Player 2 is suddenly attacked by player 1 without having any warning. Player 2 is unable to set the correct priority for messages from player 1 but player 1 can send a high urgency message to player 2.

For developing the network API, for example, the Microsoft DirectPlay API may be used. The AAI is then needed to format the output such as ACP and AOP, and may be written as a DirectPlay service provider.

Using the Microsoft DirectPlay API there are at least two different ways of extracting subscriptions. Note that the subscription should show the priority of the local participant for receiving updates about enumerated application objects. Application objects in this document are the same as "players" in the DirectPlay notation since DirectPlay "players" are application entities that can send and receive messages. A DirectPlay "player" can be controlled by a human player or it could be an autonomous game object.

In the first method information gained from DirectPlay's receive method is used. By setting the DPRECEIVE_FROMPLAYER flag and specifying the lpidFrom parameter appropriately, the method can retrieve the first message from the "player" that is identified by the lpidFrom parameter. This information can be used by the application access interface for making the subscription. If no message from the identified "player" was available in the DirectPlay message queue, it would be reasonable to put the identified application object at the top of the priority list.

In the second method the DirectPlay Send method is used, in which the idTo parameter identifies the "player" or player group that should receive the messages. It is reasonable to assume that "players" receiving frequent messages are associated with application objects with which the local player is presently interacting. The

5   AAI could hence subscribe to "players" identified by the idTo parameter.

Figure 5 shows an example of the communication stacks that may be used according to the invention. One communication stack is used for the communication between a client such as the one shown in Figure 4 and an application access server.

10   The client and the application access server comprise essentially the same type of stack. In the client the uppermost layer of the stack communicates with the application access interface layer 10 of Figure 4, and in the application access server the stack communicates with an application access server software is also shown. Figure 5 also shows a communication stack for use between an application access

15   server and another unit in the network. This other unit may be, for example, another application access server, or an application lobby server. The communication stacks used conform to the OSI model.

The client communication stack communicates with the application programming

20   interface 107 in the client. The uppermost level of the stack, is the ACP/AOP layer 109. This level is handled by the application access interface 107 in Figure 4.

From the ACP/AOP layer 109 ACP/AOP packets containing information such as application object data or subscription information are delivered to a link layer 111.

25   The link protocol may be for example PPP. In the opposite direction, the ACP/AOP layer 109 removes header information from information packets received from the application access server and forwards the application state information to the client application. The ACP packets may be terminated in AAI. The subscription itself can be handled by the client itself, if it comprises functions for handling the subscrip-

30   tion.

The lowest layer is a channel layer 113, comprising channel coding and the actual physical connection.

The application access server comprises essentially the same type of stack for communication with the client: a channel layer 113' corresponds to the channel layer of the client. The channel layer 113' is connected to an AOP/ACP layer 109' through a link layer 111'.

An AOP/ACO layer 109' in the application access server communicates directly with an application access server software 115 which is arranged to handle the AOP and ACP information.

The application access server can be built with interfaces to many different client link protocols. Ideally, an application access server should be able to handle any link protocol, including UDP, TCP and RTP. RTP is a protocol developed specifically for the transmission of voice and video data.

The link protocol should be designed so that the protocol overhead on the link from the application access server to the client is kept low. This can be done, for example, by using an appropriate link protocol where IP/UDP/RTP is not used or by efficient IP/UDP/RTP header compression. The link layer should further minimize latency and provide information to the application access server on the properties of the link. Such information could include expected bandwidth, bit error rate and link latency.

An appropriate transport protocol, here called the application transport protocol, ATP should be used on the link but application object packets and application control packet messages could also be sent directly via the link protocol.

For communication between two or more application access servers, a stream of IP packets are sent from the output buffer 65 of the local application access server (see

Figure 2) to one or more remote application access server units participating in the
ongoing application. Each IP packet contains a TCP or UDP packet and the TCP or
UDP payload is an Application Transport Protocol (ATP) packet.

5   The uppermost protocol layer of the communication stack for communication be-
tween two application access servers is an AOP/ACP layer 117 similar to the one
used in the client communication stack. An application object packet might be quite
small, that is, ~ 40 bytes or less. To make the communication between two applica-
tion access servers more efficient, several application object packets are therefore
10   aggregated in an application transport protocol (ATP) layer 119. The next layer is a
TCP or UDP layer 121 and the lowermost layer is an IP layer 123, both of which
121, 123 are well known in the art. From the IP layer information packets are
transmitted to remote application access server units. The output buffer unit (65 in
Figure 2) keeps a set of sorting buffers for collecting application object packets that
15   will be the ATP payload. The structure of these buffers depends on the distribution
strategy. There might be one sorting buffer for each remote application access
server.

An application object packet may include a field listing the clients that should get
20   the update. This field is translated to a list of application access server units that
should receive the application object packet. The application access server keeps a
table that matches client numbers to application access server numbers. This means
that all relevant application access server units eventually will get the update. The
application access server units will then dispatch the application object packets to
25   their local clients. A simple optional mode of operation is that all application access
server units receive all application data.

The local application access server identifies application object packets that need to
be sent to remote clients via remote application access server units. The application
30   state receiver of the local application access server is therefore scanned and appli-
cation object packets with the new_client_data flag set are found. An application

object packet header comprises a recipient group field listing one or more client addresses to which the application object packet should be sent. This recipient group field is examined. Client addresses are translated to remote application access server addresses and copies of the application object packet are put into the sorting buffers

5   that correspond to the recipient application access server units. The recipient group field is modified for each application object packet copy so that only recipient clients that belong to the receiving application access server or group of application access servers remain. The new_client_data flag is reset.

10   Application Object Packets contain payload from the game application. The application access server system cannot read the internal payload format of the games. Such messages are therefore wrapped in an application object packet (AOP). The header of an AOP can be read by the application access server system. It is used for appending information that is needed for the timely delivery of the game payload.

15

In the embodiment discussed here, three non-standard protocols are used in addition to standard Internet protocols such as IP, TCP, UDP and RTP:

- Application Object Packet (AOP) is a container for game data. All data that is sent from a game application to a remote game client or server is wrapped in an

20   AOP so that it can be handled by the application access server system.

- Application Control Protocol (ACP) is used for sending control messages. Control messages are sent between application access server units, clients and application lobby servers.

- Application Transport Protocol (ATP) is used for sending aggregated game data

25   between application access server units and optionally also between application access servers and clients.

Application Object Packets contain messages from the game application. The application access server system can not read the internal message format of the games.

30   Such messages are therefore wrapped in an application object packet (AOP). The

header of an AOP can be read by the application access server system. It is used for appending information that is needed for the timely delivery of the game payload.

Application messages can completely define the state of an application object or they can alternatively describe the application object relative to a reference state. AOPs are therefore of two types: Reference packet (BP) and Incremental Packet (IP).

An application object packet consists of a header followed by game specific payload: Header fields that may be used in the AOP include:

1) Application object number

2) Time stamp showing the time in the game when the AOP was generated

3) Optional packet number. Combined with the object number and the time stamp it creates a unique identifier of the packet. Packet numbers are only used if several AOPs, belonging to the same game object, have the same time stamp.

4) If, for example, voice and/or video information may be transmitted, a flag showing if the packet contains this type of information may be included.

5) A flag showing if the AOP is a reference packet or an incremental packet

6) If the AOP is incremental, a pointer to the reference AOP. This pointer could consist of the time stamp and the packet number of the reference AOP. The reference AOP could be either a basic AOP or an incremental AOP.

7) A record describing the clients that should receive the message. This can be done by listing the clients or use a predefined client group. The default is that all clients receive the data. An urgency field is associated with each receiving client or client group. This field is used for alerting the receiver if the message is urgent. Three urgency bits would be sufficient.

Flags that can be set in the urgency field are,

Forbidden: the AOP should not distributed to the client or client group

Fair_Play: use fair play mode (see below)

Very_Urgent: overrides client priorities

5   Urgent: client will be alerted

Normal: delivered according to client priorities

Not_Urgent: delivered by best effort

A simple format could be:

<number of entries> <client 1><urgency for client 1><client 2><urgency for

10   client 2> ......

Later items in the list overrides previous items. The field:

2, all Forbidden, client_3 Urgent

could e.g. mean that all clients are forbidden to receive the AOP with the ex-

ception of client number 3 that will get the AOP in the Urgent mode.

15   8) Size of payload.

The application control protocol (ACP) is used for sending control messages be-

tween game clients, application access server units and the application lobby server

(ALS). An outline of ACP is provided here. Each ACP packet consists of a header

20   and a message body.

An ACP message may include the following fields:

1) ACP message type

2) Time stamp showing the time in the game when the message was generated

25   3) Size of message

4) Message body

Note that the source of the message is identified by higher protocol levels. In the

following the messages that can be transmitted using ACP will be outlined.

ACP messages from client to the application access server may include the following:

- Terminate client. The application access server management system removes the client from all records and notifies the Application Lobby Server (ALS). The client is responsible for notifying the ALS if it is leaving the game. The ALS is responsible for finalizing the contact with the terminated client e.g. by sending the final scores and notifying other players.

- Subscriptions

- Add game object. A new game object number is generated by the application access server and an application access server memory is allocated for receiving object state information from the client.

- Remove game object. The object is removed from all application access server memory after all clients and remote application access server units that are listed for receiving updates have received the last update.

- Send estimated latencies for objects. This request includes a list of object numbers. The application access server responds by sending the estimated end-to-end latencies for the objects. The client application uses the estimates for latency hiding.

- Define object group. Object groups are useful for giving short names to long lists of object numbers that otherwise might have to be sent on the client link. Direct-Play handles hierarchical "player" groups so the game API will be able to provide useful group definitions. The application access server stores object group information and treats an object group as an alias for a list of objects. The message might have the format:

    message type = define object group; > <object group name> <list of objects>

- Define client group. Client groups are useful for setting short names on long lists of client numbers that otherwise might have to be sent over the client link in AOP fields. The application access server stores client group information and treats a client group as an alias for a list of clients. The message might have the

format.

message type = define client group <client group name> <list of clients>

- Send time reference. This message is used for downloading a reference time from the application access server.

ACP messages from the application access server to the client include the following:

- Urgency list. This message is used for alerting the client if urgent unread AOPs are waiting. The application access server scans all unread AOPs that has the present client on the AOP recipient list. The urgency list could have the format.

  <urgency class 1> <list of object numbers> <urgency class 2> <list of object numbers> and so on

- Latency estimate. This consists of a set of entries according to:

  <game object number> <upstream latency> <upstream latency variance> <downstream latency> <downstream latency variance>

  A "don't know" symbol can be used for any field except the first.

- Confirm object group number. The application access server has received a "define object group" message from a client. It confirms that a global object group number has been allocated. The message body would include.

  <global object group number> <client's object group name>

  A simple method for allocating object numbers would be:

  Assume that a total of N application access server units are active. Enumerate all active application access server units. If application access server number k is asked for a new object number it allocates the lowest free object number from the series {k, N+k, 2N+k, 3N+k, ...}

- Confirm client group number. The application access server has received a "define client group message" from a client. It confirms that a global client group number has been allocated. The message body would include

  <global client group number> <clients client group name>

The same algorithm that is used for allocating global object group names could be used for allocating client group names.

- Clock synchronization. The application access server sends a time reference according to:

<client time> = <time> + <client link latency>

The client has an algorithm for adjusting the local clock according to a series of received clock synchronization messages.

ACP messages from one application access server to another include:

- Aggregated subscriptions. Aggregated priority lists showing the objects that the clients belonging to the sending application access server need to see, that is, the objects for which information should be sent to the sending application access server. The list has the same format as a simple client priority list. It is built by adding all valid local client priority lists and removing duplicates.

- Resend request. This message has the same format as a simple priority list and is interpreted as a request for resending the state of the listed objects.

- Define object group. Object group definitions may be distributed between application access servers. Using global names for long lists of object numbers helps in reducing traffic.

- Define client group. Client group definitions may be distributed between application access server.

ACP messages from an application access server to an application lobby server include:

- Client has been terminated. This message is sent when a client has voluntarily disconnected from the game.

- Client time-out The application access server may send a message to the ALS if a client has been silent for a long time or if the link to the client is closed. The ALS decides on further actions such as removing the client from the game.

An ALS can act as a client. The ALS might e.g. control game objects that keep the scores of the game. A port of the ALS is then connected to the AAS as a client using a "link" protocol such as TCP/IP. The ALS can therefore use the same messages as a client. Other ALS to AAS messages are listed here:

Application set-up information, including

- Network address including port number identifying the ALS
- An URL that uniquely identifies the game
- IP addresses of other application access server units that participates in the game
- A list of network addresses including application port numbers that identifies the clients.
- Optionally a list of enumerated game objects. For each object it is specified which application access server that is responsible for storing the state. It might also be specified who is authorized to update the state.
- Optionally, an initial game state to be stored.
- Optional data specifying control strategies such as the time-outs.
- Add application access server. The network address, number and client list of the new application access server is specified.
- Remove application access server. The number of the application access server to be removed is specified
- Add client. This includes the network address of the new client; optionally the initial state of the new game objects that are controlled by the new client.
- Remove client. The application access server system removes the client from the game.
- Modified client status. This message changes the clients authority to control game objects or associates the client to a different application access server. The format could be:

  <client number> <application access server number> <object numbers>

  where the object numbers indicates game objects that are controlled by the client.

The Application Transport Protocol (ATP) is used for transporting Application

Object Packets (AOP) and Application Control Protocol (ACP) messages. The ATP

has a series of AOPs and ACP messages in the payload and is mainly used for

sending aggregated data between application access server units. Application access

5   server units would typically communicate using the protocol stack IP/UDP/ATP.

ATP is hence on the same logical level as the RTP protocol.

The ATP header includes:

- An URL that identifies the game

10   - Earliest AOP or ACP time stamp

- Latest AOP or ACP time stamp

- Number of AOPs and ACP messages

ATP is normally only used for communication between application access servers,

15   as shown. If the bandwidth of the connection between a client and an application

server is sufficiently high it may also be used for these connections.   ·

Conventional protocols such as TCP/IP and/or UDP/IP are used for communication

with the Application Lobby Server and other application access server units.

20   TCP/IP should be used for setting up the game and UDP/IP should be used for

sending real-time game data, for the following reasons:

TCP resending and reordering is too slow and complex for the game application.

TCP reordering might delay the delivery of the most recent update for the purpose

of delivering too old data. Resending of data that is no longer needed is obviously

25   wasteful. The application access server units know the precise priorities and can re-

quest resending from several sources if needed.

RTP can be used for sending time stamps but RTP is designed for audio and video

streams and is not very well suited for game streams. RTP could, however, be used

30   for carrying voice and video streams that are associated with games.

A "fair play mode" may also be implemented in the system. In this mode the application access server units synchronize the delivery of critical application information so those updates are received by all clients concerned simultaneously. This mode can be used for competitions.

The Application Lobby Server decides, while setting up an application, if the fair play mode is available. If the fair play mode is allowed, the sending client decides for each transmitted AOP if fair play delivery should be used. This is done by setting the Fair_Play flag in the urgency field of the AOP. The application access server system is now responsible for delivering the AOP "simultaneously" to all clients. This requirement overrides all other client priorities.

A possible technical solution for this is the "bucket synchronization" method. The application access server units agree to update the clients with a fixed absolute delay. Updates arriving earlier have to wait for the agreed time slot. This method has the disadvantage of increasing the overall delay in the system. The fair play mode can alternatively be handled by the game application. In this mode the client application applies the agreed delay to received game data using the time stamp of each game object to calculate how much the packet should be delayed to achieve the agreed absolute delay.

34

Figure 6 shows another embodiment of an application access server 150 according to the invention. Where nothing else is said, this embodiment is similar to the one discussed in the above. In this embodiment the communication functions of the application access server are divided between one or more application routers 152 and

5   an application server 154. The application server 154 comprises an application state record 155 and a client priority list 161 as described in connection with Figure 1. The application router 152 communicates with one or more clients 170, through the client's application programming interface API 172, which will be discussed in more detail below. Each application access server 150 comprises one application

10   server 154 serving one or more application routers 152. The application router 152 also communicates with other nodes in the network, such as application routers in other application access servers (not shown). Preferably, the application server 154 and the application router 152 are implemented as separate hardware units to optimize each with regard to their specific functions. Alternatively, they may have sepa-

15   rate processor resources allocated to them. In particular, the application router must run without interruption since interruptions would affect negatively the real-time performance of the client applications.

As before, the application access server to which a client is connected is that client's

20   local application access server, and the client is a local client of that application access server. Similarly, the application server and the application router serving the client are the client's local application server and application router.

The application server receives ATP packets containing basic objects (see below)

25   and control messages from clients through the application router. Before processing a basic object e.g. by putting it in the application state database, the application server checks that the sending client is allowed to update the object. If the client is not authorised to update the object, the object is rejected and the sending client may receive an error message. Before processing a control message, the application server

30   checks that the sending client is allowed to request the indicated operation, and if

to the sending client. Consider e.g. a case when the sending client requests that an

object group is deleted. This group might have been created by the content server,

which is a special client, responsible for managing the game state and

communicating with the application access server over a standard client API. The

5   content server has uploaded authorisation rules to the game access server that

forbids clients to delete the object group. In this case, the application server will

reject the control message that requests the deletion of the object group.

This authorisation check is performed using tables of authorisation rules that shows

10   what kind of operations that the client is allowed to perform. The authorisation

tables are set up by clients or by the control units using initialization files.

A client can, if authorised to do so, request the creation of an object group. A client

can also add members, remove members and delete the object group. Members of

15   the an object group are objects and object groups. Request for any such operation

on object groups are sent by the client as a control message which is broadcasted by

the application routers to all application servers that participate in the session. The

object group is identified by an identifier which has the same format as an object

identifier. Each application server includes an object group database 162 that for

20   each application session stores the object groups that has been created during the

session. The object group database 162 stores at least the following fields for each

object group: 1) the object group identifier 2) a list of the members of the object

group 3) a list of the parents of the object group. A parent of an object group A is an

object group that has the object group A as a member. The object database is

25   organised as the object database in the previous embodiment but for each object

there is a list of object groups that are parents to the object.

Objects received from clients, and from other application access servers. that should

form part of the application state is forwarded from the application router 152 to the

30   application server 154 and stored in the application state record 155. Subscription

information received from clients and other application access servers is forwarded to the application server 154 and stored in the client priority list 161.

A site manager 174 controls the function of one or more application access servers 150, each comprising one or more application routers 152 and an application server 154, preferably through a connection to the application router 152. The site manager could also use the standard protocol SNMP (Simple Network Management Protocol) to control the site.

One site therefore comprises one site manager and a number of application access servers 150 controlled by one site manager 174. The main tasks of the site manager 174 are connection management, client connections, resource management, configuration of application servers and routers, admission control, authentication of clients, time synchronisation for a site, network management and supervision of load and status of application servers and application routers.

The site manager optionally receives Application Transport Protocol (ATP) packets containing control messages from clients. It should be noted that the ATP in this embodiment is different from the ATP described in connection with the previous embodiment and will be described below. Before processing a control message, the SM checks that the sending client is allowed to request the indicated operation, if not the control message is rejected and the sending client receives an optional error message. This authorisation check is performed using tables of authorisation rules that shows what kind of operations that the client is allowed to. The authorisation tables are set up by clients or by a lobby access server178, using initialization files.

One or more lobby access servers 178 are provided for connection to the application lobby server 176, one or more lobby access servers 178. The application lobby server is discussed in connection with Fig. 2. The lobby access server or servers 178 are connected to the site manager 174. The lobby access server handles game session requests from lobby servers and arranges and manages game sessions. This in-

cludes registration and authentication of lobby servers, and accounting. It also keeps
information about games and lobby severs in a database. A central control unit (not
shown) may be used to supervise and maintain the application communication sys-
tem.

5

An application communication system comprises one or more interconnected sites
and lobby access servers. A central control unit (not shown) may be used to super-
vise and maintain the application communication system.

10   The lobby access server 178 may comprise or be connected to a database 180 com-
prising information about the application lobby servers present in the network, such
as which application lobby server or servers are allowed to add clients to a particu-
lar game, or a particular application access server. The application lobby server 176
may comprise, or be connected to, a database 182 comprising client data such as
15   authorization profiles for each client.

The communication between the site manager and the application server or servers
it controls includes adding and removing sessions, and adding and removing clients
from sessions. The communication between the site manager and the application
20   router or routers includes the same as above as well as login permissions for clients.
Also, error messages are sent from the application server or application router to the
site manager.

Preferably, the local application router 152 can handle both stream objects and basic
25   objects received from clients, other application routers or other units in the network.
Clients send one copy of each update of a basic object to the local application ac-
cess server. The object contains e.g. the status of the player's avatar. The basic ob-
ject is forwarded to all relevant application servers in the session. Basic objects
received by an application router from clients or other application access servers are
30   stored in the application state record of the associated application server. Players
that want to receive the stream object issues a subscription to the local application

server. The application server sends a series of updates of the object from the local application state record according to the parameters of the subscription. Stream objects are not stored in the application server. Basic objects and stream objects will be discussed in more detail in the following. An application router receives Application Transport Protocol (ATP) packets from clients, the local application server, from other application routers and optionally from the local site manager. ATP packets are transported through other network protocols, such as UDP/IP. Other application routers are either remote (belonging to a remote Application Access Server) or local (belonging to the same Application Access Server as the application router). The received ATP packets are of three different types: basic objects and stream objects in Application Object Packets (AOP), control data in Application Control Packets (ACP), and client messages in Client Message Packets (CMP). ATP packets are either deleted or routed by unicast or multicast to receiving clients, application routers, the local application server or to the local site manager. The different routing cases are shown in the table. Control packets that are addressed to the receiving application router are terminated and not routed.

A routing table for an application router is shown in the table below:

Received ATP packet	Routed to the following nodes	Manda-tory(M)/ Optional(O)	Comment
Stream object from client	Local clients that are subscribing to the stream object	M	
	Other application routers that has clients that are subscribing to the stream object	M	
	All other application routers	O	A special case that is easy to imple-ment
	Clients (if any) that are specified in the optional address field of the AOP. Remote clients are reached via their local application routers.	M	Direct addressing requires more bandwidth than subscriptions and should be used sparingly.

Basic object from client	The local application server	M	
	Clients (if any) that are specified in the optional address field of the AOP. Remote clients are reached via their local application routers.	M	
Client message from client	Clients that are specified in the address field of the CMP. Remote clients are reached via their local application routers.	M	
Control packet from client	The local application server	O	Depending on the nature of the message
	The local application router	O	
	The local site manager	O	
Basic object from local application server	Clients that are specified in the address field of the AOP.	M	
	Remote application servers (via one of their application routers) that are specified in the address field of the AOP.	M	
	All remote application servers optionally via a multicast address	O	A special case that is easy to implement
	Remote clients that have subscribed to the object (via their application router) that are specified in the address field of the AOP.	O	A special case that is expensive in bandwidth but makes the game state well defined at any given time
Control packet from local application server	Clients, local site manager, remote application server or application router that are specified in the address field of the ACP.		
Stream object from application router	Local clients that are subscribing to the stream object	M	
	Clients and application	M	

	routers (if any) that are specified in the optional address field of the AOP. Remote clients are reached via their local application routers.		
Basic object from application router	The local application server	M	
	Clients (if any) that are specified in the optional address field of the AOP.	M	
	Remote application servers (via one of their application routers) that are specified in the optional address field of the AOP.	M	
Control packet from application router	Clients, local site manager, remote application server or application router that are specified in the address field of the ACP.	M	
Control packet from the local site manager.	Clients, local or remote application server or application router that are specified in the address field of the ACP.	M	

Stream objects are not stored in the application server. Clients typically use them for sending the position of moving items in a virtual world. They can be transported

5   very fast through the network. Only the most recent version of a stream object is assumed to matter for the receiving client. The handling of stream objects is typically unreliable, as they may be dropped deliberately by application routers for flow control. All stream objects received by an application router are forwarded to all local clients that have subscribed to the stream object, or to a stream object group com-

10   prising the stream object. The application router 152 also comprises priority or subscription information for stream objects, since these objects are not handled by the application server, as will be discussed below.

Typically a client who wants to join an application session, for a example a particular ongoing multiplayer session, connects over the Internet to a application

5   lobby server 176. The application lobby server provides information on ongoing application sessions and usually also allows clients to create new sessions and invite other clients to join. A client may send a request to the application lobby server 176 to create, delete, join or leave an application session. If the application lobby server accepts any such session control request it will forward the request to a lobby access

10   server 178. If creation of a session is requested the lobby access server 178 will respond by issuing a application session identifier. For requests such as a request to delete a session or add or remove a client the application lobby server 176 will include the application session identifier in the request that is sent to the lobby access server 178. The lobby access server 178 responds by investigating if

15   resources are available in the application access server system to match the requested operation. If resources are available, the lobby access server 178 will perform the requested operation.

If creation of a session is requested the lobby access server 178 will select a set of

20   sites that will serve the session and send control messages including the session identifier to the relevant site managers 174 requesting that the session is initiated on suitable application access servers 150.

If deletion of a session is requested the lobby access server 178 will send control

25   messages including the session identifier to the site managers 174 of the sites that are running the session requesting that the session is deleted.

If joining of a client is requested the application lobby server 176 includes at least the network address of the client 170, the session identifier and optionally a site

30   identifier in the control message to the lobby access server 178. The network address might e.g. be the IP address and the port number of the client application

process. The lobby access server 178 selects a suitable site and sends a control message to the site manager 174 of the site requesting that the client 170 joins the game session. This control message includes at least the network address of the client 170 and the session identifier to the site manager 174 of that site. The site

5   manager selects a suitable application access server and a suitable application router within that unit to be responsible for serving the client and sends a control message to the client 170 inviting the client to join the game. The network address of the application router that will serve the client and a password is included in the message. The client 170 contacts the relevant application access server 150

10   including the password in the control message. The application access server 150 includes the client 170 in the local client database and the client can proceed with sending and receiving game data.

Alternatively, the site manager 174 can send the network address of the application

15   router 152 that will serve the client 170 to the lobby access server 178, the lobby access server 178 will forward the network address and a password to the application lobby server 150. The application lobby server 176 will forward the network address and a password to the requesting client 170 that connects to the relevant application access server as described above.

20

       Although no buffers, software processes etc. are shown in Figure 6, it will be appreciated by the skilled person that they may be needed to implement the application access server of the invention.

25   Clients 170, application access servers 150 and site managers 174 communicate using the Application Transport Protocol (ATP). ATP packets are typically carried as payload in UDP packets as discussed in the previous embodiment. The ATP used in this embodiment will be described in the following. The Application Transport Protocol includes two protocol levels 1) compound ATP packet 2) regular ATP

30   packet.

A compound ATP packet comprises a of a Source Header and several regular ATP packets. The Source Header comprises the following fields:

1) Application session identifier

2) Client identifier indicating the sending client

5 3) Optional fields that are used by the protocol for guaranteed transmission including acknowledgement fields and packet counters.

A regular ATP packet (in the following also called "ATP packet") consists of an ATP header, an ATP optional header, an optional ATP Target Header and an ATP

10 Content Packet.

The ATP header may include the following fields:

1) A Type Field, which is a set of flags indicating the type of the Content Packet and the presence of optional fields in the Content Packet. Content packet types

15 are, control message, client message, basic object or stream object. The intended receiver of a message can be indicated in the Type Field.

2) A flag indicating if the ATP packet is sent in the reliable or the unreliable mode

3) A flag indicating if a Target Header is present

4) A set of flags indicating the presence and content of the ATP optional header.

20 5) A field indicating the size of the content packet

The ATP optional header consists of the following optional fields:

1) session identifier

2) client identifier

25 3) object identifier

The ATP optional header is used for identifying the Content Packet. The application payload that is associated with a basic object might e.g. be sent in the Content Packet. The ATP optional header is used for identifying the the basic object using

30 the relative addressing system that is described below.

The ATP target header TH is used for direct addressing of ATP packets. The ATP address of the receiver is indicated in the ATP target header. The first position in the TH is a byte that holds the size of the TH. The TH consists of an array of dynamic address fields that is either a list of client identifiers indicating the intended receivers of the message or a list of stream object keys. A key is an attribute of the stream object that may be set by the sender. Clients can subscribe to stream objects that carry a specific key without knowing the object identifiers of the wanted stream objects. In a game like Quake, keys can be used for selecting stream objects belonging to different rooms. The stream objects are used for sending the positions of the avatars in the game. A key is assigned to each room in the game. Players that are sending stream objects add the key that corresponds to the room where the avatar is located. Players that are receiving data subscribe to stream objects with keys that correspond to rooms in which they are interested. Stream objectkeys can alternatively be used to identify a team or to classify objects, for example, as dangerous. A stream object filter in the local application access server receives the player's subscriptions for stream object keys and ensures that the player gets stream objects with the requested keys.

The ATP Content Packet (CP) can be of any of the following types 1) basic object 2) stream object 3) control message 4) client message. A basic object, stream object and client message CP includes application payload. Control messages include a message type and message parameters. The Aplication Control Protocol specifies the format of the message type field and the parameters of each message.

Session identifiers, client identifiers and object identifiers are dynamic address fields as described below. Client groups and object groups have the same identifier format as clients and objects, respectively.

The following names and definitions are used in this document. An Application Object Packet is an ATP packet that contains an application object in the Content Packet. An Application Control Protocol packet is an ATP packet that contains a

control message in the content packet. Sending a basic object, stream object or client message means always that an ATP packet with the appropriate type of content packet is sent. A client message packet is an ATP packet containing a receiver address and a client message in the content packet.

5

ATP usage examples include the following:

A) A client is sending a basic object to the local application server. The object identifier but not the client identifier or the session identifier are put in the ATP optional header. No ATP target header is needed. The object payload is put in

10   the content packet. The use of optional headers and dynamic address fields means that the header overhead can be as small as four bytes. This is essential since objects are often sent over a low bandwidth connection.

B) A client is sending a client message to another client. The object identifier and the client identifier of the sending client are put in the ATP optional header. No

15   session identifier is needed. A target header is used containing the client identifier of the receiving client. The message payload is put in the content packet. The input filter of the application router on the sending side adds the session identifier to the target header. The output filter of the application router on the receiving side removes the session identifier before forwarding the packet

20   to the receiving client.

ATP uses dynamic address fields. A dynamic field is a variable length prefix code. A prefix code is a code where each code field in a bit stream can be uniquely decoded without reference to previous code words. A simple example is the alphabet

25   { 10, 110, 1110, ... }. Huffman codes are optimal (shortest expected length) prefix codes for a given statistical distribution of symbols.

An alternative format of dynamic address fields is dynamic size integers. Dynamic size integers are well known in computer science. Dynamic size integers are used whenever small values are most common, but large values must be possible. For 2-

30   byte dynamic size integer, the most significant bit tells the size of the integer – 1 means one byte and 0 means 2 bytes. For 4 byte dynamic size integers, the two most

significant bits tells the size of the integer – 01 means one byte, 10 means two

bytes, 11 means 3 bytes and 00 means 4 bytes.

Dynamic address fields may be used according to one of the following two

5   methods:

In the first method, ATP has fields for three different identifiers 1) application

session identifier 2) client identifier and 3) object identifier. The application session

identifier is globally unique and allocated by a central authority such as the net

manager or lobby access server. Each client, application router, application server

10   and optionally site managers has a client identifier and are in the following called

nodes. A client group identifier has the format of a client identifier.

The client identifier is unique only in a specific application session. A central

authority such as the or lobby access server or a central control unit allocates client

15   identifiers. The client identifier identifies the node only if the applicaction session

identifier is known. ATP stores the client identifier in a dynamic field. Each node in

an application session is allocated a client identifier with as few bits as possible.

Frequently used client identifiers may be allocated the shortest codes e.g. according

to the Huffman coding procedure.

20

The object identifier is unique to a given client in a given session. The client

allocates the object identifier. The object identifier identifies the object only if the

session identifier and client identifier is known. ATP stores the object identifier in a

dynamic field. Each object is allocated an object identifier with as few bits as

25   possible. Frequently used object identifiers may be allocated the shortest codes e.g.

according to the Huffman coding procedure.

The second method is the same as the first. with the following exception. The client

identifier consist of two dynamic fields. The first field is the identifier of the

30   application router that is serving the client. The second dynamic field is a client

index. This index is selected to be as short as possible and is unique to the client

given that the application session identifier and local application router client identifier is known. The client identifier of the local application server can alternatively be used instead of the client identifier of the local application router.

5   The advantage of using dynamic address fields and relative addressing as described above is that the bandwidth usage and delays are minimised. Only the relevant address fields are sent. Examples of this are:

1) A client participates only in one application session. ATP packets containing objects from the clients contain only the object identifier. The application ses-

10   sion identifier and the client identifier are not sent, since the application router knows this implicitly.

2) ATP packets containing objects that are sent from the local application server to the client need only to include the object identifier and the client identifier of the client that owns the object.

15   3) A client that creates a new object can immediately allocate an object identifier. Central allocation of globally unique object identifiers would require communication with the central authority before a new object can be created.

The client group addressing is handled according to the following: A client group is

20   identified in ATP by a client identifier. The identifier is created using the same method as for other client identifiers. Creation of a client group is requested by a client during the application session or the creation of the client group is performed as a part of the initialization of the session.

25   Each ATP packet includes a Type Field in the ATP header. The Type Field indicates the nature of the content. Different codes in the type field indicate e.g. that the content is basic object, stream object, control packet or a client message. The application router can use the type field for default routing in some cases without reading and analysing the address fields. Examples of default routing based on the

30   type field are:

1) all basic objects are routed to the local application server 2) control packets with one specific code in the type field are routed to the local application server and control packets with one other specific code in the type field are routed to the local site manager.

Figure 7 shows an application router 152 used in the embodiment discussed in connection with Figure 6, and how ATP packets flow through the application router. The application router 152 consists of a set of input filters 190, 192, a router core 194 and a set of output filters 196, 198. The application router has an input and an output filter for each node with which it communicates. The router core 194 comprises a number of routing tables 199 for performing the routing of ATP packets.

Packets come from three types of sources – from the application server of the same application access server, and other similar units, e.g. the site manager, from other application routers and from application clients connected to this application router. There are different routing tables 199 for all three types. There is one incoming filter 190, 192 and one outgoing filter 196, 198 for every connection to another application router or an application client. For the connections to the application server, no input or output filters are needed. The filters implement assembling several application packets into transport packages and also handles resending for reliable packets. The application client output 196, 198 filters remove redundant information on the client line and also supplies load balancing for the downlink to the application client 170.

The client 170 that is associated with the output filter, or any other client that is authorised, can send a subscription for a stream object or a stream object key to the output filter of the receiving client. The subscription is sent as a control message that indicates the content of the subscription and the client identifier of the receiving client. The output filter will scan stream objects that arrive to the application routers. Stream objects matching the subscription are sent to the client, possibly

after applying a drop rate. A stream object matches a subscription either if the object identifier of the stream object is equal to the object identifier that is given in the subscription or if a stream object key that is carried by the stream object matches a stream object key that is given in the subscription.

5

Consider a case when the client (that is associated with the output filter) has subscribed to a stream object and the stream object is received from the sending client. The client or any other client that is authorised can set a droprate that is associated with a given stream object. This droprate is sent in a control message to

10   the local application router. If the droprate is set to a value $0 \leq R \leq 1$ the output filter will deliberatly delete a fraction R of the stream objects. If e.g. R = 0.9 only 10% of the arriving packets will be allowed to pass down to the client.

The output filter will delete ATP header fields that do not need to be sent to the

15   client. A direct addressed message from another client contains e.g. the application session identifier and the client identifier of the receiving client in the header. This information is not useful to the receiving client and is therefore deleted by the application router output filter.

20   The input filter for a specific node, such as a client, receives ATP packets containing stream objects, basic objects, control messages and direct addressed client messages from the node. Before sending the ATP packets to the routing core the following operations is performed:

1) The input filter checks if the client is allowed to update a stream object or a

25   basic object or if the sending client is allowed to send a direct addressed message to the client that is indicated in the message. This authorisation check is performed using authorisation tables that shows what kind of operations that the client is allowed to perform. The authorisation tables are set up by clients or by the LAS via the site manager using initialization files.

2) The input filter adds the address of the sending client in the appropriate ATP field. This address is not sent on the communication link between the sending client and the application router in order to save bandwidth

3) The application router stores lists of client groups. Such lists associates a client
5   identifier that indicates a client group to a list of client identifiers that indicates clients. Lists of client groups are set up by clients or by the LAS via the site manager. If an ATP packet with a client group in the reciever address fields is received by the input filter, it will do one of the following three alternative operations: A) Find the list of client identifiers that belongs to the client group.
10   Find the number N of application routers that should receive at least one copy of the ATP packet. Make N copies of the ATP packet that is put in the output filter to each of the receiving application router. There is a flag in the ATP header that indicates that the content should not be distributed to any remote receiver but only to local clients. This flag is set in each output packet. This is the preferred
15   mode of operation.

4) Find the list of client identifiers that belongs to the client group. Find the number N of application routers that should receive at least one copy of the ATP packet. Make N copies of the ATP packet that is put in the output filter to each of the receiving application router. In each packet replace the client group
20   identifier with a list of receiving clients where the list includes only members of the client group that are local clients to the receiving application router. Duplicate the ATP packet so that in each duplicate the client group identifier is replaced with one unique client identifier indicating one of the clients in the client identifier list that is associated with the client group.

25 5) A default header is stored by the input filter for each stream object that the client that is associated with the input filter is sending. This default header contains the stream object keys associated with the stream object. If the client sends an explicit header containing stream object keys this header will replace the default header that is stored by the input filter. If the client sends the stream object
30   without a header containing stream object keys, the input filter will add the default header to the stream object before sending it to the router core. This

means that the sending client do not have to send one copy of the stream object keys with each copy of the stream object. It is sufficient to send a header containing the stream object keys each time that the keys are changing and the first time that the stream object is sent.

5

A regular ATP packet carries a content that is either an application object (stream or basic), a control message or a client message. A compound ATP packet consists of an optional small compound header and at least one and usually several regular ATP packets. A compund ATP packet is sent as payload in a standard transport

10   protocol. Usually compound ATP packets are sent as payload in UDP packets.

The method for collecting regular ATP packets in compound ATP packets is essential for achieving the correct balance between bandwidth efficiency and low latency. Large compound ATP packets gives high transmission latency but low

15   header overhead and hence efficient bandwidth utilisation. Aggregation of regular ATP packets into compound ATP packets takes place in several modules of the system.

For aggregation in the application client for sending to the local application router,

20   two alternative methods may be used: In the first method, aggregation is under direct control of the application program. Regular ATP packets are sent to an output buffer. The application program decides when the content of the buffer is put into a compound ATP packet and is transmitted to the local application router. In the second method, the aggregation is performed by an automatic algorithm in the

25   client. Regular ATP packets are sent to an output buffer by the application program. The algorithm decides when the content of the buffer is put into a compound ATP packet and is transmitted to the local application router. An example of a suitable algorithm is: A compound packet is sent if the buffer size exceeds a given size S. A compound packet is sent if the time since the last transmission of a compound

30   packet exceeds a given time interval T. The parameters S and T are set by the application program.

For aggregation in application router for sending to a client, for example, the same algorithm as in the second method above may be used.

For aggregation in application router for sending to a application router, sending the
5   maximum size ATP compound packet that is allowed by the next protocol level e.g. UDP will usually work well, since the communication link between application routers can be assumed to be very fast and regular packets are sent very frequently.

If the bandwidth is constrained or ATP packets are sent infrequently it is advised to
10   use the same algorithm as in method 2 of the previous section on aggregations in the application client.

The size S (in bits) should be selected as $S < V*T0$ where V is the communication speed of the link between the two application routers measured in bits per second
15   and T0 is the allowed transmission latency usually 1-10ms. The time interval T should be in the intervall 1-10ms.

Clients can send subscriptions for basic objects as control messages to the local application server. A subscription specifies the receiving clients, a target for the
20   subscription and subscription parameters. A subscription is an instruction to the application server to send a series of basic objects to the receiving clients. A series of updates of each basic object is sent according to the parameters of the subscription and depending on how the source of the basic object updates it.

25   The client sending the subscription is usually the receiving client but a client may also make a subscription on behalf of another client. A subscription can also be issed on behalf of a client group. All clients in the client group will then receive the result of the subscription.

30   The target for the subscription is a set of basic objects identified by object identifiers according to the ATP protocol. The target can be described by 1) an

object identifier, 2) a set of object identifier 3) an object group identifier 4) a mixed set of object identifiers and object group identifiers 5) general Boolean expression including object groups and objects as operands and logical operators AND, OR and NOT. A correctly defined target expression is evaluated by the application

5   server using the information in the object database and the object group database. The result of this evaluation is always a finite set of basic objects.

Subscription parameters may be used to specify how often. how many times, and for how long the content of the subscription is to be delivered. This may be

10   specified as the maximum rate at which updated versions of the object should be received. If the owner of the object updates it at a slower rate. each new version will be received as soon as it is delivered. If the object is updated faster than the selected maximum rate, the updates will be received at precisely the selected rate. Some of the intermediate versions will not be received in this case.

15

A content server or master client may have superior knowledge of the state of the game. It could therefore be suitable that the master client subscribes to basic objects on behalf of a target client. The master client then builds and submits the subscription to the system. The local application access server delivers the basic

20   objects requested in the subscription directly to the target client or target client group.

A subscription belongs to a client or a client group. A subscription refers to a set of game objects. This set can be expressed as a general Boolean expression including

25   object groups and objects.

The Application Access Server (AAS) will try to deliver objects with the subscription frequency. The delivery time is the inverse of the subscription frequency. Objects that exist in the database when the subscription is issued will be

30   delivered within the delivery time. If an object in the database is modified while the subscription is active. The AAS will try to deliver it within the delivery time.

A Subscription Class is a positive integer. Two methods for delivering data according to the subscription class is described here.

5   Method 1: Subscriptions with the highest class are first satisfied completely. Next, the AAS will try to satisfy the subscriptions in the next highest class completely. This means that subscriptions in lower classes never might be satisfied. If the bandwidth is insufficient for delivering objects within a subscription class with the requested frequency the AAS will scale all frequencies within the class so that it

10   becomes feasible to deliver to the client.

Method 2: Three different classes are used. The different classes are called 1 = HIGH, 2 = MEDIUM and 3 = LOW. With class 1 the object is always sent to the subscriber at maximum rate until the buffer is full and the rate drops to zero. With

15   class 2 the object's send rate to the subscriber is proportional to buffer load. The less buffer capacity the lower rate. With class 3, the object is sent to the subscriber only if there are no other objects in the buffer.

In one specific implementation application object packets may include several enu-

20   merated payloads. In that case subscriptions have an optional Weight parameter. The Weight parameter is an integer. Weight = n means that the first n payloads in the object are delivered.

Some objects may not be updated at regular intervals, but instead an exact number

25   of times. Therefore, it is possible to request a specific number of updates regarding a particular object. The application server will then set a counter that is associated with the object to an integer value that is specified in the subscription. The counter is decremented by one unit each time that the object is delivered. The subscription is removed if Counter = 0.

30

The client may request that the object be sent at the requested frequency even if the same update has been sent before. This is done by setting the "forced sending" FS flag.

5   Also, to prevent information relating to old subscriptions to overload the network, the time to live, or duration of a subscription should be set. The subscription is removed after the subscription time-to-live expires.

Stream objects are distributed between application routers using one of the

10   following methods:

Method 1) A stream object that is received by one application router is duplicated and sent to all other application routers that participates in the application session. This may be done efficiently, for example, by associating all application routers that particpates in the application session with an IP multicast address.

15   Method 2) Each application router receives a set of subscriptions to stream objects and stream object keys from local clients. All such subscriptions are aggregated to a joint subscription that includes all stream objects and stream object keys that local clients subscribes to without any duplications. Stream objects but not stream object keys that are generated by local clients are removed from the aggregated

20   subscription. The aggregated subscription is sent to all other application routers. A application router that receives an aggregated subscription from another application router and that finds the source to any of the stream objects and stream object keys in the aggregrated subscription is among the local clients will send that part of the subscription to the subscribing application router.

25   Method 3) Each application router prepares an aggregated set of subscriptions as in Method 2. The aggregated subscription is sent to a parent application router that has been assigned by the site manager. The parent application router delivers the subset of the aggregated subscription that is available and includes the remainder in its own aggregated subscription that is sent to the parent application router that has

30   been assigned to the first parent application router. This assumes that the

management system has organised an hierarchy of application routers for the application session.

Basic objects are distributed between application servers. A application server is
always reached via one of its associated local application routers. When an
application server sends a basic object to another application server, the first
application server always sends the basic object with the second application server's
identifier in the recipient address field of the ATP packet to one of the local
application routers. This application router will send the object to a local application
router of the second application server. The local application router of the second
application server will send the object to the second application server.

Basic objects are distributed between application servers using one of the following
methods:

Method 1) A basic object that is received by one application server is duplicated
and sent to all other application servers that participates in the application session.
This may optionally be done efficiently by associating one local application router
for each application server that particpates in the application session with an IP
multicast address.

Method 2) Each application server receives a set of subscriptions to basic objects
from local clients. All such subscriptions are aggregated to a joint subscription that
includes all basic objects that local clients subscribes to but without any
subscription parameters. Basic objects that are generated by local clients are
removed from the aggregated subscription. The aggregated subscription is sent to
all other application servers. A application server that receives an aggregated
subscription from another application server and that finds the source of any of the
basic objects in the aggregated subscription is among the local clients will
immediatly send such objects to the subscribing application server and store the
relevant part of the subscription. Any further updates of the relevant objects will
also be sent immediatly to the subscribing application server.

Method 3) Each application server prepares an aggregated set of subscriptions as in
Method 2. The aggregated subscription is sent to a parent application server that has
been assigned by the site manager. The parent application router delivers the subset
of the aggregated subscription that is available and includes the remainder in its

5   own aggregated subscription that is sent to the parent application server that has
been assigned to the first parent application server. This assumes that the
management system has organised a hierarchy of application servers for the
application session.

10   Control data that needs to be distributed among all application access servers that
participates in a session includes e.g. clients joining and leaving and creation and
destruction of client groups. Such control data should be multicasted to all
application routers in the session. application routers will either terminate the
control message or forward it to the local application server or site manager as

15   required.

Some clients may be registered with an application access server as spectators.
Spectators are not allowed to upload application actions but they can upload a sub-
scription that depends on the spectator's point of view in the game.

20

Application access servers according to the invention may also be used in hierarchi-
cal systems. Figure 8 shows an example of such a configuration in which an higher-
level application access server 201 serves other application access server units 203,
205, 207, which in turn serve application clients 209. The higher-level application

25   access server 201 receives ATP packets with game data and aggregated subscrip-
tions from downstream application access server units that are considered to be
"clients". The higher-level application access server unit 201 communicates with
other peer application access server units and possibly with application access
server units that are on a still higher level in the hierarchy. Hierarchical application

30   access server systems are used for creating a hierarchical distributed representation
of the game state for an application involving a large number of clients.

Remote application access server units might be organized in a multicasting tree and in that case there will be one sorting buffer per multicasting group. A simple distribution strategy where all information is sent to all remote application access server units may be used. In that case there is only one buffer.

A set of application clients could have broadband connections providing low latency access so that no application access server is needed. A typical example is that all players are on the same LAN. If the game application and API still expects an application access server, it would be possible to run an application access server function in software on the client machine.

A application access server unit could also be situated at any intermediate position in the network. This means that the "link" protocol between client and application access server could optionally be UDP/IP or TCP/IP. Alternatively, a central game server running a dial-in game service could use application access server units as components. The central site would then consist of a set of modem pools that are connected to application access server units and application lobby servers. This would be an efficient and scaleable architecture for building a central site but it would not have the advantage of early aggregation of game traffic.

Claims

1. A server unit for use in a communications network, said server unit comprising
receiving means for receiving information from at least a first node, each first node

5   being represented as a client unit, said information comprising at least part of the
state information about a distributed interactive application, said server unit being
characterized in that it comprises

- state information storing means for storing application state information com-
prised in the information received through the receiving means from said first

10   node,

- first transmission means for transmitting the at least part of the information from
the state information storing means to at least one second node in the network,

- second transmission means for transmitting at least part of the information stored
in said state information storing means to said at least first node.

15

2. A server unit according to claim 1, further comprising a client priority list for
storing priority information received from said at least one first node through the re-
ceiving means, said priority information identifying objects of the distributed inter-
active application about which a first client at said first node wishes receive infor-

20   mation and identify the preferred order in which the first node wishes to receive the
objects, the second transmission means being adapted to transmit information from
said state information storing means to said at least one client unit in dependence of
the priority information received from that first node.

25   3. A server unit according to claim 1, further comprising a subscription list for stor-
ing subscription information received from said at least one first node through the
receiving means, said subscription information identifying how often information
related to a certain object is to be received in the first node.

4. A server unit according to claim 1 or 3, further comprising a subscription list for storing subscription information received from said at least one first node through the receiving means, said subscription information identifying the number of times information related to a certain object is to be received in the first node.

5

5. A server unit according to any one of the preceding claims, wherein the receiving means is adapted to receive priority information relating to said first node from at least a third node, being represented as a client node, and forward or not forward application state information originating in said third node to at least said first

10   node in dependence of said urgency information.

6. A server unit according to any one of the preceding claims, wherein the receiving means is adapted to receive priority information relating to said first node from at least a third node, and inform said first node about the priority information.

15

7. A server unit according to any one of the preceding claims, wherein the state information storing means is adapted to discard stored state information after a predetermined period of time.

20   8. A server unit according to any one of the preceding claims, wherein the information transmitted to said at least first node is state information received from at least one other node.

9. A server unit according to any one of the preceding claims, wherein said state in-

25   formation storing means is adapted to store the application state information in terms of properties of objects, said objects constituting the application state.

10. A server unit according to claim 9, wherein said state information storing means is adapted to store reference packets or datagrams describing the state of an object at

30   a given time.

11. A server unit according to claim 10, wherein said state information storing means is adapted to store incremental packets describing the change of the state of the object since a particular previous packet, and information identifying said previous packet.

12. A server unit according to any one of the preceding claims, further comprising registration means for registering authorization information for said at least one first node, said authorization information indicating the application state objects that said first node is authorized to change.

13. A server unit according to any one of the preceding claims, comprising
at least one application router comprising the first and second transmission means and the receiving means,
an application server comprising the information storing means.

14. A server unit according to claim 13, wherein the application server and said at least one application router are implemented as separate hardware units.

15. A server unit according to claim 13, wherein the application server and said at least one application router are implemented in software, the software relating to the application server and the at least one application router, respectively, having reserved computing resources.

16 A server unit according to any one of the claims 13-15, wherein the application router is arranged to receive a stream object and transmit it to at least one other application access server.

17. A server unit according to claim 14, wherein the application router is arranged to broadcast the stream object to all other application access servers involved in a communication session.

5   18. A server unit according to any one of the claims 13-17, wherein the application router is arranged to receive a basic object and forward it to the at least one application server for storage in a database in the application server and to receive an addressed data packet relating to a basic object from the application server and forward it to a client and/or at least one other server unit.

10

19. A server unit according to any one of the claims 13-18, wherein the application router is arranged to receive control information from the application server and forward it to a client and/or at least one other server unit, and/or a site manager.

15   20. A client unit for use in a terminal in a communications network, comprising application software for a distributed interactive application, said client unit being **characterized** in that it comprises
        - at least one input means for reading an input from said terminal, said input constituting at least application state information for the distributed interactive appli-
20   cation,
        - transmission means for transmitting application state information to an application access server in the communications network,
        - receiving means for receiving at least application state information for the distributed interactive application from said application access server
25   - means for displaying said state information received by said receiving means.

21. A client unit according to claim 20, further comprising priority setting means for setting priority information for specifying at least one other application state object about which available information should be received as soon as possible and

transmitting means for transmitting said priority information from said priority set-

ting means to said application access server.

22. A client unit according to claim 20 or 21, further comprising priority data means

for setting priority data for specifying at least one other node that should receive

state information from the client as soon as possible.

23. A client unit according to any one of the claims 20-22, wherein said transmis-

sion means for transmitting application state information is adapted to arrange in-

formation in application object packets, each packet relating to one object consti-

tuting part of the application, before transmitting the packets to the access server,

and said receiving means is adapted to extract information about application objects

from packets received from the access server.

24. A communications network characterized in that it comprises at least one server

unit according to any one of the claims 1 to 19.

25. A communications network according to claim 24, further comprising at least

one client unit according to any one of the claims 20 to 23.

26. A communications network according to any one of claims 24-25, further com-

prising at least a second application access server and connection means connecting

said first and second application access servers, wherein communication resources

may be reserved on said connection means.

27. A communications network according to any one of claims 24-26, further com-

prising at least a third application access server. communicating with said first ap-

plication access server but not with said second application access server. in such a

way that a hierarchy of application access servers is achieved in the network.

28. A communications network according to any one of the claims 24-27, further comprising at least one site manager arranged to control at least one application access server.

5   29. A communications network according to claim 28, further comprising a lobby access server arranged to constitute an interface between at least one application lobby server and said at least one site manager.

1/4

FIG.1

FIG.2

2 /4

AOP 11
AOP 12
AOP 13
AOP21
AOP 22
AOP23
AOP24
AOP 31
AOP41
AOP42

AOS1 { (AOP 11, AOP 12, AOP 13)

AOS2 { (AOP21, AOP 22, AOP23, AOP24)

AOS3 { (AOP 31)

AOS4 { (AOP41, AOP42)
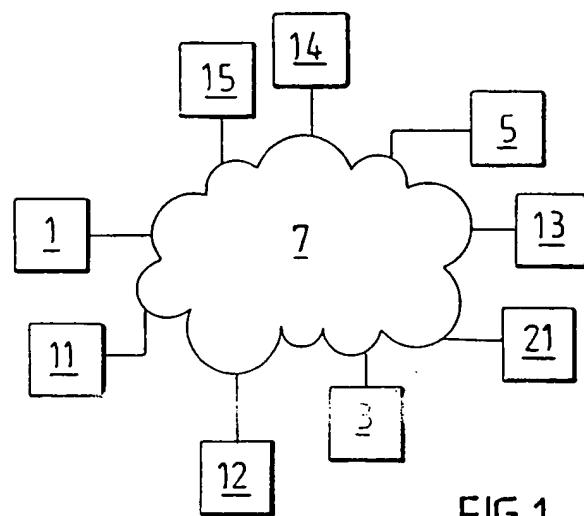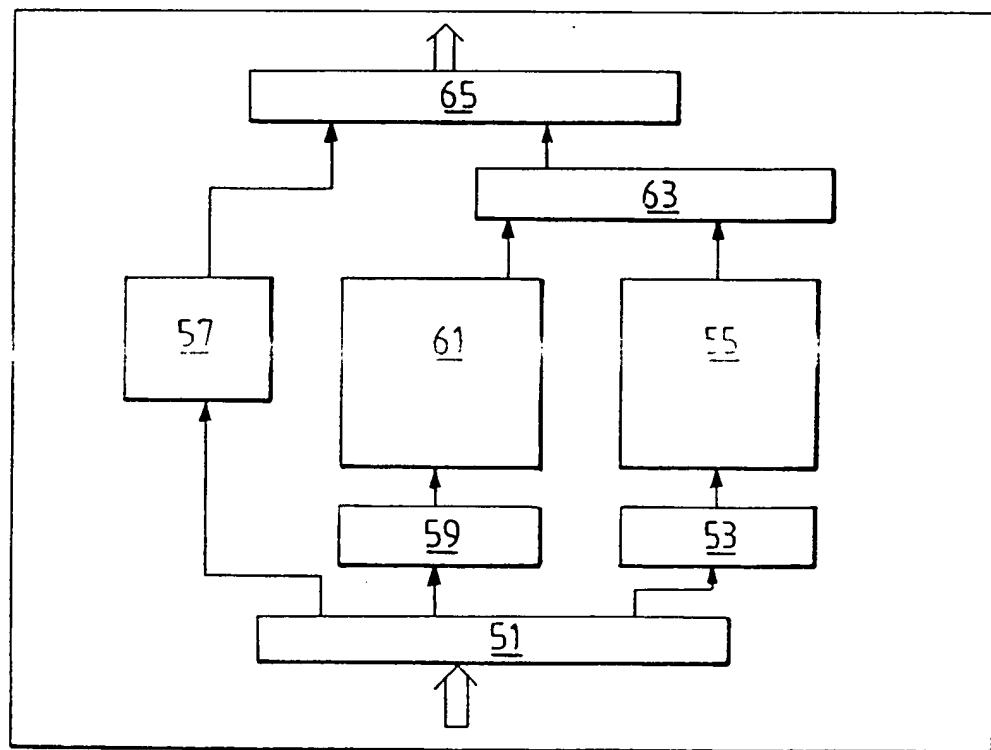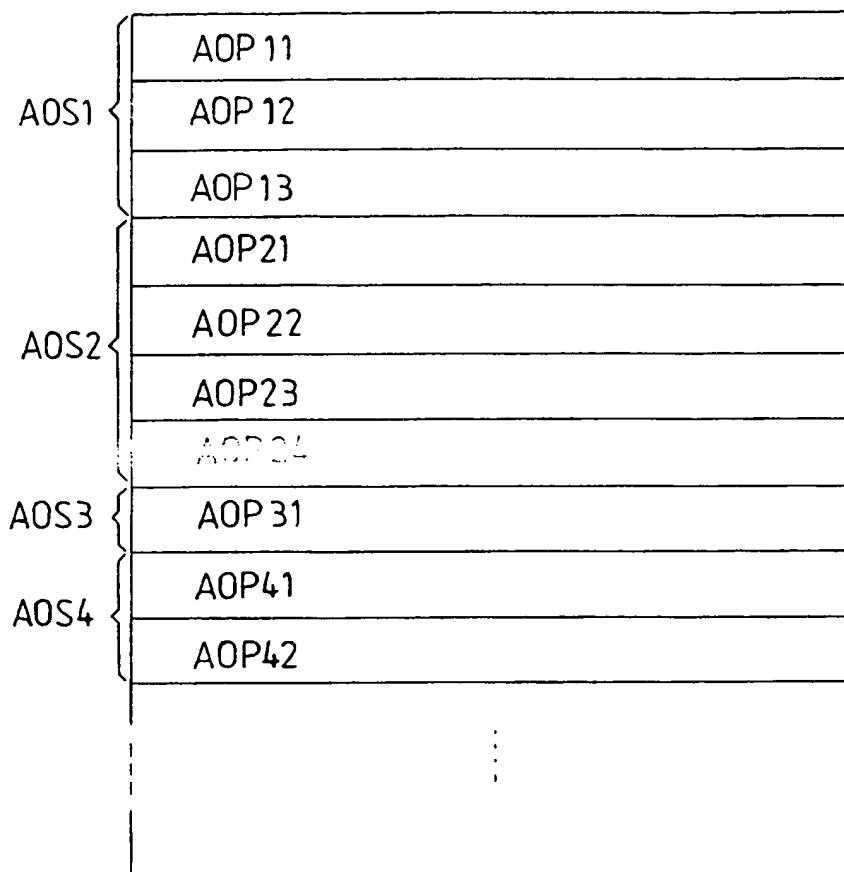
FIG. 3
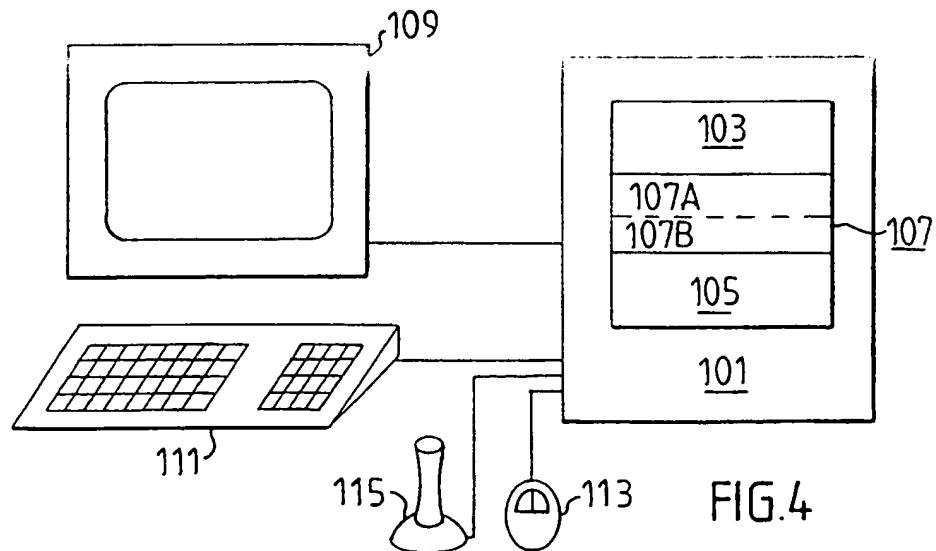
FIG.4

3/4

FIG.5

FIG.8

4/4

FIG.6

FIG. 7
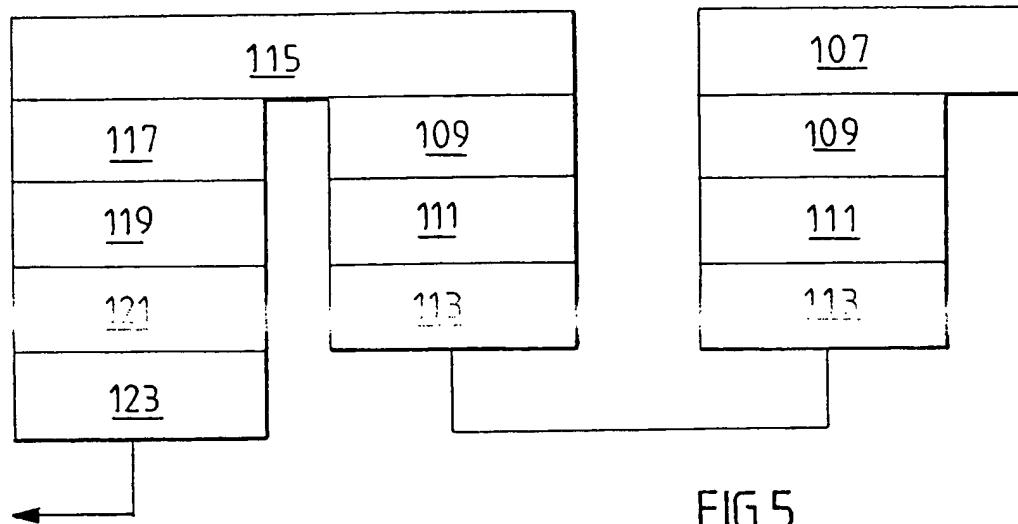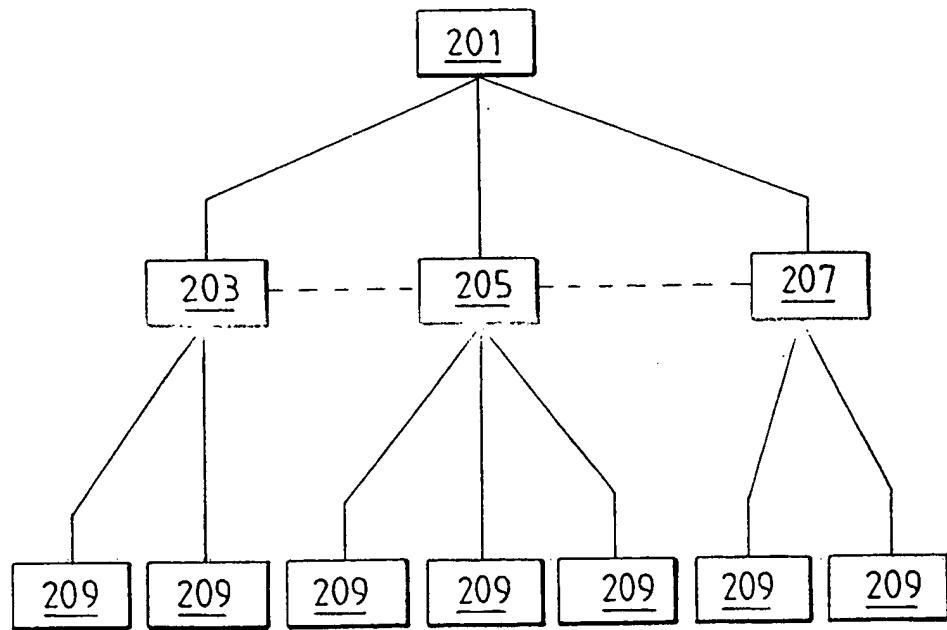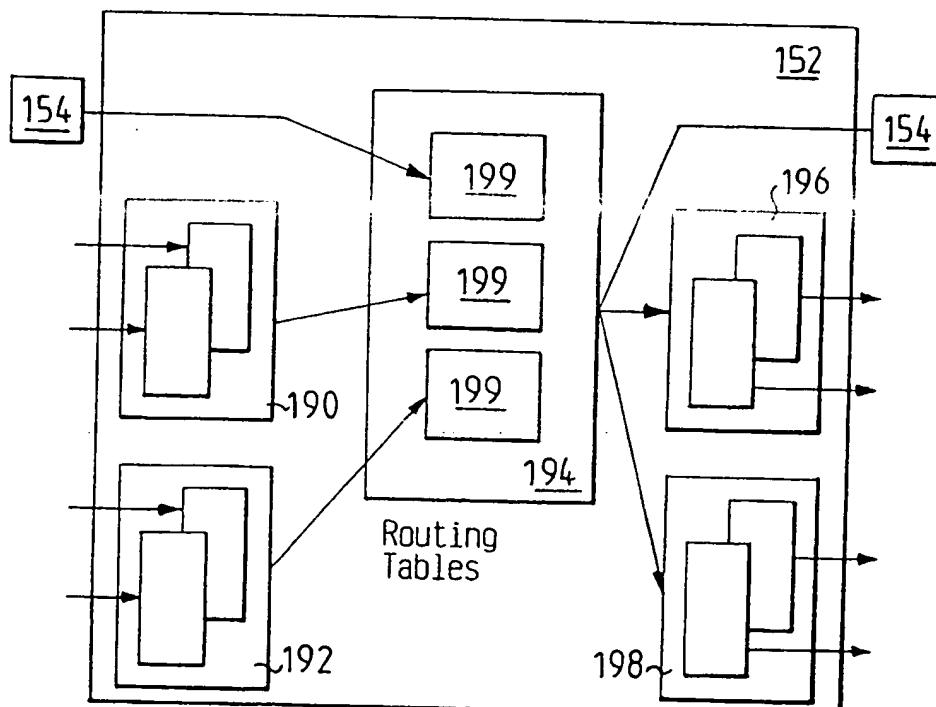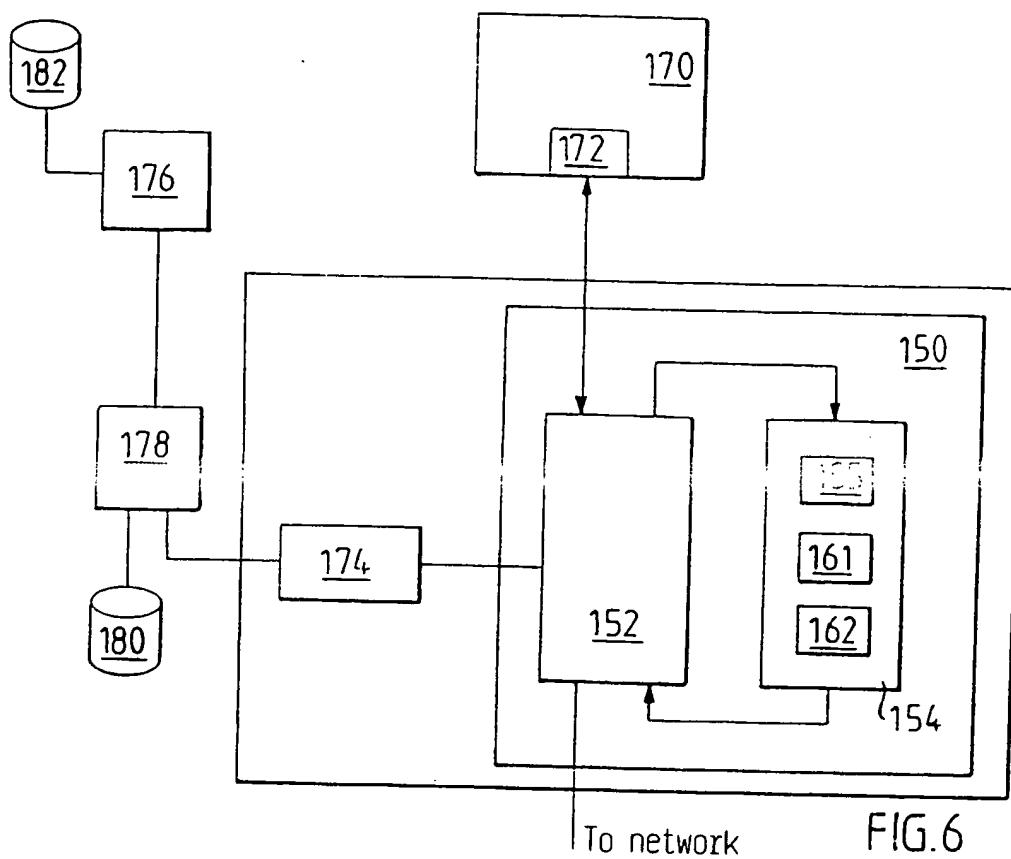
INTERNATIONAL SEARCH REPORT

	International application No.
	PCT/SE 00/00932

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: G06F 19/00 // G06F161:00
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5890963 A (WEI YEN), 6 April 1999 (06.04.99), column 3, line 48 - line 61; column 7, line 36 - line 42; column 8, line 27 - line 33, claim 20, abstract	1-29
	--	
X	US 5899810 A (JOSHUA E. SMITH), 4 May 1999 (04.05.99), column 1 - column 6, figure 1	1-29
	--	
X	WO 9814897 A2 (PHILIPS ELECTRONICS N.V. ET AL), 9 April 1998 (09.04.98), page 1, line 22 - line 24; page 2, line 5 - line 11; page 2, line 24 - page 4, line 16	1,20
A		2-19,21-29
	--	

[X] Further documents are listed in the continuation of Box C.　　　[X] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier document but published on or after the international filing date	"X"	document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)		
"O"	document referring to an oral disclosure, use, exhibition or other means	"Y"	document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
15 Sept 2000	20 -09- 2000
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Jesper Bergstrand/LR Telephone No. +46 8 782 25 00

INTERNATIONAL SEARCH REPORT

International application No.

PCT/SE 00/00932

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0909069 A2 (MITSUBISHI DENKI KABUSHIKI KAISHA), 14 April 1999 (14.04.99), column 1, line 12 - line 35; column 8, line 53 - line 57, figure 4, claim 1	1,20
A		2-19,21-29
	--	
A	WO 9531061 A1 (CATAPULT ENTERTAINMENT, INC.), 16 November 1995 (16.11.95), see the whole document	1-29
	--	
A	US 5846132 A (WILLIAM W. JUNKIN), 8 December 1998 (08.12.98), see the whole document	1-29
	--	

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
US	5890963 A	06/04/99	NONE		
US	5899810 A	04/05/99	AU WO	6244098 A 9832507 A	18/08/98 30/07/98
WO	9814897 A2	09/04/98	EP JP US	0876653 A 2000502478 T 6025801 A	11/11/98 29/02/00 15/02/00
EP	0909069 A2	14/04/99	JP US	11161622 A 6006254 A	18/06/99 21/12/99
WO	9531061 A1	16/11/95	AU EP US	2471995 A 0774186 A 5956485 A	29/11/95 21/05/97 21/09/99
US	5846132 A	08/12/98	AU WO	2610597 A 9737734 A	29/10/97 16/10/97